REMARKS

Claims 1, 9, 10, and 14 have been amended above. No new matter has been added. Claims 1, 3, and 9 to 15 are pending in the present application. In view of this Response, Applicant respectfully requests reconsideration of and allowance of the present application.

Interview

Applicant thanks Examiner Timblin for the courtesies extended to Applicant's representative Wesley Jones during the telephone interview of August 6, 2007. A summary of the substance of the interview is set forth below.

During the interview, Applicant's representative asserted that the applied §103 references did not render obvious independent claims 1, 9 and 10. Applicant's representative also maintained that the pointers disclosed by *Kodovalla et al.* are used to link pages of records and are not used to link individual records. No agreement as to the patentability of the claims was reached. Examiner Timblin suggested that claims 1 and 9 be amended to clarify that steps (e) and (f) of each claim are directed to storing a pointer with a previous saved version of a record (i.e., a previously retrieved and stored version of a record) that points to a newly saved version of the same record. Examiner Timblin stated that he would review the amended claims with his supervisor.

Prior Art Rejections

Claims 1, 3, 9 to 15 Define over Kodavalla et al. and Paul et al.

Claims 1, 3, and 9 to 15 were rejected under 35 U.S.C. § 103(a) as being unpatentable over *Kodavalla et al.* (U.S. Patent No. 5,717,919) in view of *Paul et al.* (U.S. Patent No. 7,051,080). Applicant respectfully requests withdrawal of these rejections because neither *Kodavalla et al.* nor *Paul et al.*, either alone or in combination, teaches or suggests all elements of independent claims 1, 9 and 10.

Representative claim 1 recites:

> A method for managing bufferpages and **redundant copies of records** in a local memory associated with a mobile device application, comprising:
>
> (a) retrieving a first record from a remote database memory in response to a request from a first recordset;
>
> (b) saving the first record on a first bufferpage of the local memory associated with the mobile device application, the first bufferpage being associated with the first recordset;
>
> (c) repeating steps (a) and (b) for at least one further record;
>
> (d) when a next record requested by the first recordset is larger than a freespace on the first bufferpage, saving the next record on a second bufferpage of the local memory associated with the mobile device application, the second bufferpage being associated with the first recordset;
>
> (e) **determining if one of the first record, the at least one further record, and the next record was previously retrieved and saved** in the local memory associated with the mobile device application by at least one of the first recordset and at least one second recordset **as one of a prior saved version of the first record, a prior saved version of the at least one further record, and a prior saved version of the next record, respectively**; and
>
> (f) **storing a pointer with one of the prior saved version of the first record, the prior saved version of the at least one further record, and the prior saved version of the next record, the pointer pointing to the one of the first record, the at least one further record, and the next record if one of the first record, the at least one further record, and the next record was previously retrieved and saved as one of the prior saved version of the first record, the prior saved version of the at least one further record, and the prior saved version of the next record, respectively, otherwise creating a business object kernel pointing to one of the first record, the at least one further record, and the next record**.

Kodavalla et al. does not teach or suggest "determining if one of the first record, the at least one further record, and the next record was previously retrieved and saved . . . as one of a prior saved version of the first record, a prior saved version of the at least one further record, and a prior saved version of the next record, respectively" as recited in claim 1. Further, *Kodavalla et al.* does not teach or suggest "storing a pointer with the prior record . . . pointing to the one of the first record, the at least one further record, and the next record if one of the first record, the

at least one further record, and the next record was previously retrieved and saved as one of the prior saved version of the first record, the prior saved version of the at least one further record, and the prior saved version of the next record, respectively," – and does not teach or suggest "creating a business object kernel pointing to one of the first record, the at least one further record, and the next record" if one of the first record, the at least one further record, and the next record was not previously retrieved and saved as prior version of itself, as recited in claim 1.

The present invention provides for the efficient management of databuffers and redundant copies of records in a mobile application. As described in the present application, a record can be stored to an active bufferpage provided the active bufferpage has room to store the record (*e.g.*, see specification text concerning Figure 11). If the record is larger than the freespace of the active bufferpage, then the record can be stored on a next active bufferpage. To manage possible redundant copies of the record (*i.e.*, prior saved versions of the same record), the present invention may determine if the record was previously retrieved and stored (possibly on a previous bufferpage), as recited in claim 1. If it is determined that a previous version of the same record has already been stored, then a pointer can be stored with this previously stored version of the record that points to the newly stored record. In this way, reliance on old records can be avoided as an application that attempts to use the previously saved version of the record will be directed to the more current version of the record.

In contrast to the present invention, *Kodavalla et al.* fails to teach or disclose at least the management of redundant copies of individual records as claimed in the present application. Instead, *Kodavalla et al.* is directed to the storing of data records on data pages that are linked together by pointers. In *Kodavalla et al.*, data page chains are described at column 7, lines 29-42, as follows:

> As shown in FIG. 3A, the data records or rows of a database table

> are actually stored in a particular structure known as a data page. A data page may be viewed as a storage unit (*e.g.*, 2K storage block) which holds one or more records, such as page 310. When a data page is “full," typically on the order of about 50 to 100 records, it is necessary to allocate a new data page. Every page which is allocated is linked to its previous and next neighboring pages via forward and backward page pointers (*e.g.*, pointers 311, 321), so that logically a chain (linked list) of (physical) pages exists. This forms the "page chain," such as the page chain 300 shown in FIG. 3A. Typically, identifiers or "Page IDs" for the first and last page of a page chain are maintained in a system catalog.

As shown, *Kodavalla et al.* describes the use of pointers to link data pages that store multiple records and fails to disclose the use of pointers to link different versions of the same individual record. *Kodovalla et al.* clearly states that page chains are used to increase the speed at which multiple process can insert records and fails to ever mention how the page chains are used to manage redundant copies of records (*See* Abstract). In addition, *Kodavalla et al.* fails to teach or disclose, after storing a record, determining if a prior version of the same record has been previously stored. In addition, *Kodavalla et al.* fails to teach or disclose storing a pointer from a determined previously stored record to the newly stored version of the record. Accordingly, *Kodavalla et al.* fails to teach or disclose at least the above identified features of claim 1. *Paul et al.* is directed to interacting with client processes on a mobile device and is not directed to managing bufferpages and redundant copies of records and so fails to cure the deficiencies of *Kodavalla et al.* Therefore, Applicant requests that the rejection of claim 1 be reconsidered and withdrawn as *Kodavalla et al.* and *Paul et al.* fail to teach or suggest all elements of claim 1.

Claims 3 and 11 depend from independent claim 1 and are allowable for at least the reasons applicable to claim 1, as well as due to the features recited therein.

Independent claims 9 and 10 recite limitations similar to those of claim 1. Accordingly, claims 9 and 10 are allowable over *Kodavalla et al.* and *Paul et al.* for at least those reasons

mentioned above with respect to claim 1.

Claim 12 depends from claim 9 and claims 13 to 15 depend from claim 10 and are allowable for at least the reasons applicable to claims from which they respectively depend, as well as due to the features recited therein.

Applicant notes in particular that *Kodovalla et al.* clearly fails to teach or suggest the features of dependent claims 13 and 15 as alleged in the Office Action. Regarding claim 13, the Office Action has failed to show how *Kodovalla et al.* teaches or suggests the use of a lookup table to determine if a record has been previously retrieved and saved. None of the cites to *Kodovalla et al.* provided by the Office Action state or otherwise indicate that a lookup table is used to perform this operation.

Further, the Office Action has alleged that the function "SL_CHGVALUE" presented at col. 56 of *Kodovalla et al.* teaches or suggests the features of claim 15. Applicant respectfully disagrees. This portion of *Kodovalla et al.* appears to have been cited as disclosing the features of claim 15 because the word "count" is mentioned within the provided programming code, even though the "count" used in this function is a measure of the allocations to a page chain, not the number of times a specific, individual record has been retrieved and saved. (*See* col. 56, lines following *SLC_LASTPG_PLUS*: "We are updating the pageid of the last page as well as incrementing the counter that tracks the number of allocations to this page chain"). Clearly this reference to a "count" does not teach or disclose a counter as claimed in claim 15.

CONCLUSION

Applicant respectfully requests entry of the above amendments and favorable action in connection with this application. The Office is hereby authorized to charge any additional fees or credit any overpayments under 37 C.F.R. 1.16 or 1.17 to Kenyon & Kenyon Deposit Account No. 11-0600. The Examiner is invited to contact the undersigned at (202) 220-4419 to discuss any matter concerning this application.

Applicant respectfully submits that all claims are allowable and requests allowance of the present application.

Respectfully submitted,

KENYON & KENYON LLP

Dated: August 8, 2007

By: /Wesley W. Jones/

Wesley W. Jones
(Reg. No. 56,552)

KENYON & KENYON LLP
1500 K Street, NW, Suite 700
Washington, DC 20005-1257
Telephone: (202) 220-4419
Facsimile: (202) 220-2201
670336_2.DOC